Wavelet Based Interactive Video Communication and Image Database Consulting

# Wavelet Transforms Using the Lifting Scheme

*Geert Uytterhoeven* *Dirk Roose*
*Adhemar Bultheel*

Report ITA-Wavelets-WP1.1 (Revised version), April 28, 1997

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Wavelet Based Interactive Video Communication and Image Database Consulting

# Wavelet Transforms Using the Lifting Scheme


*Geert Uytterhoeven* *Dirk Roose*
*Adhemar Bultheel*



Report ITA-Wavelets-WP1.1 (Revised version), April 28, 1997



Department of Computer Science, K.U.Leuven



**Abstract**

This report briefly reviews terminology relevant in wavelet transforms. The Lifting Scheme for biorthogonal CDF wavelets of type $(m,n)$, $m,n = 1,\dots,6$ is described and their integer transforms are given in detail.



**Acknowledgements**

This research is supported by the Flemish Information Technology Action Program ('Vlaams Actieprogramma Informatietechnologie'), project number ITA/950244.

# Contents


**1 Introduction** **2**

**2 The Wavelet Transform** **2**

2.1 The Basic Transform Step . . . 2
2.2 Wavelet and Scaling Functions . . . 3
2.3 Example: Biorthogonal Cohen-Daubechies-Feauveau . . . 3
2.4 Multi-Resolution . . . 7
2.5 The Mathematics behind the Wavelet Transform . . . 7
2.6 Properties of Wavelets . . . 8
2.7 Higher Dimensions . . . 9
2.8 A simple compression example . . . 9

**3 The Lifting Scheme** **9**

3.1 Introduction . . . 9
3.2 Algorithm . . . 12
3.3 Example: Biorthogonal Cohen-Daubechies-Feauveau . . . 13
3.3.1 CDF $(1, x)$ . . . 13
3.3.2 CDF $(2, x)$ . . . 13
3.3.3 CDF $(3, x)$ . . . 14
3.3.4 CDF $(4, x)$ . . . 15
3.3.5 CDF $(5, x)$ . . . 16
3.3.6 CDF $(6, x)$ . . . 16
3.4 The integer wavelet transform . . . 17
3.5 Bounds of wavelet coefficients and bit growth . . . 19

# 1 Introduction

This report is mainly intended as an internal working report of the IWT project "Wavelet Based Interactive Video Communication and Image Database Consulting". It serves several purposes.

First it wants to give a catalog of wavelet related terminology. Thus a first part of the report quickly reviews the theory of the wavelet transform in one and two variables.

The compression properties of the biorthogonal Cohen-Daubechies-Feauveau (CDF) wavelets are illustrated with a simple compression technique.

Next we discuss the Lifting Scheme into more detail. This scheme is an algorithm, originally designed to compute second generation wavelets in an efficient way. Later it has been shown that it can be used to generate also the "classical" (bi)orthogonal wavelets of Cohen-Daubechies-Feauveau (CDF) type with several advantages over the classical computational schemes. More precisely, we give an inventory of the lifting steps for CDF wavelet transforms of type $(1, x)$, $(2, x)$, $\ldots$, $(6, x)$.

Classical wavelet transforms convert floating point numbers to floating point numbers. However, in many multimedia applications (images, video, audio) the input data consists of integer values only. An advantage of the Lifting Scheme is that it can be converted easily into a transform that maps integers to integers, while retaining the perfect reconstruction property. Some implementation aspects of this transform are also discussed.

# 2 The Wavelet Transform

## 2.1 The Basic Transform Step

The basic block in a wavelet transform [4] is a filter bank, consisting of 2 filters (cfr. fig. 1). A discrete signal $S$ is filtered by a high pass filter ($\widetilde{g}$) and a low pass filter ($\widetilde{h}$), and downsampled. The results are a high pass ($HP$) and a low pass ($LP$) signal, each containing half as much samples as the input signal $S$.

For the inverse transform, first the signals $HP$ and $LP$ are upsampled by putting zeroes in between every sample. After that they are filtered by the filters $g$ and $h$ and the result is added together. In the case of perfect reconstruction — we'll only consider perfect reconstruction filter banks here — the resulting signal is equal to the original signal $S$.

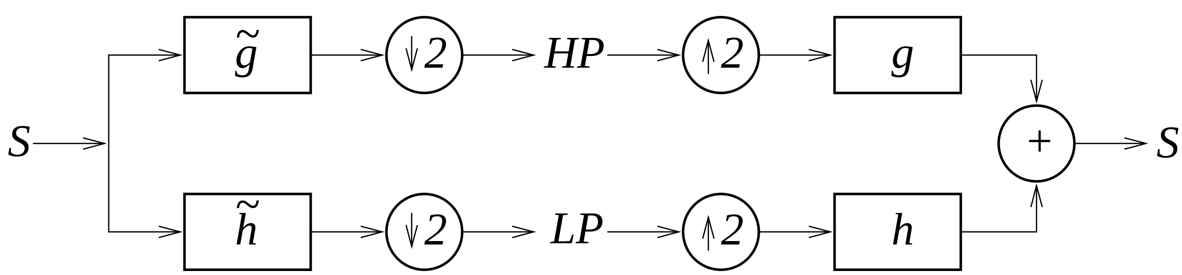



Figure 1: The filter bank algorithm.

The filters $g$, $h$, $\widetilde{g}$ and $\widetilde{h}$ are called wavelet filters if they fulfill certain conditions. The filters $\widetilde{g}$ and $\widetilde{h}$ are called dual filters w.r.t. $g$ and $h$.

### 2.2 Wavelet and Scaling Functions

The wavelet filters $g$ and $h$ uniquely define a (primal) wavelet function $\psi(x)$ and a (primal) scaling function $\varphi(x)$. The shape of these functions can easily be found be iteratively applying an upsampling and filter step on a signal sequence containing only zeros, except for one sample. Analogously, the filters $\widetilde{g}$ and $\widetilde{h}$ define a dual wavelet function $\widetilde{\psi}(x)$ and a dual scaling function $\widetilde{\varphi}(x)$.

If the filters $g$ and $\widetilde{g}$, and $h$ and $\widetilde{h}$ are equal to each other, then the primal and dual wavelet and scaling functions coincide too. In this case the wavelets are called orthogonal wavelets. In the other, more generic, case they are called biorthogonal wavelets.

The regularity of a basis function can be measured by the number of vanishing moments. A function has $n$ vanishing moments if the first $n$ moments $\mathcal{M}_j$ $(j \leq n)$

$$\mathcal{M}_j = \int_{-\infty}^{\infty} x^j f(x)\, dx$$

are zero. The more vanishing moments a function has, the smoother the function is.

The mathematical meaning is that all polynomials $x^j$ with a degree $j < n$ can be represented exactly in the basis $f_i(x)$ derived from the function $f(x)$:

$$x^j = \sum_i a_i f_i(x).$$

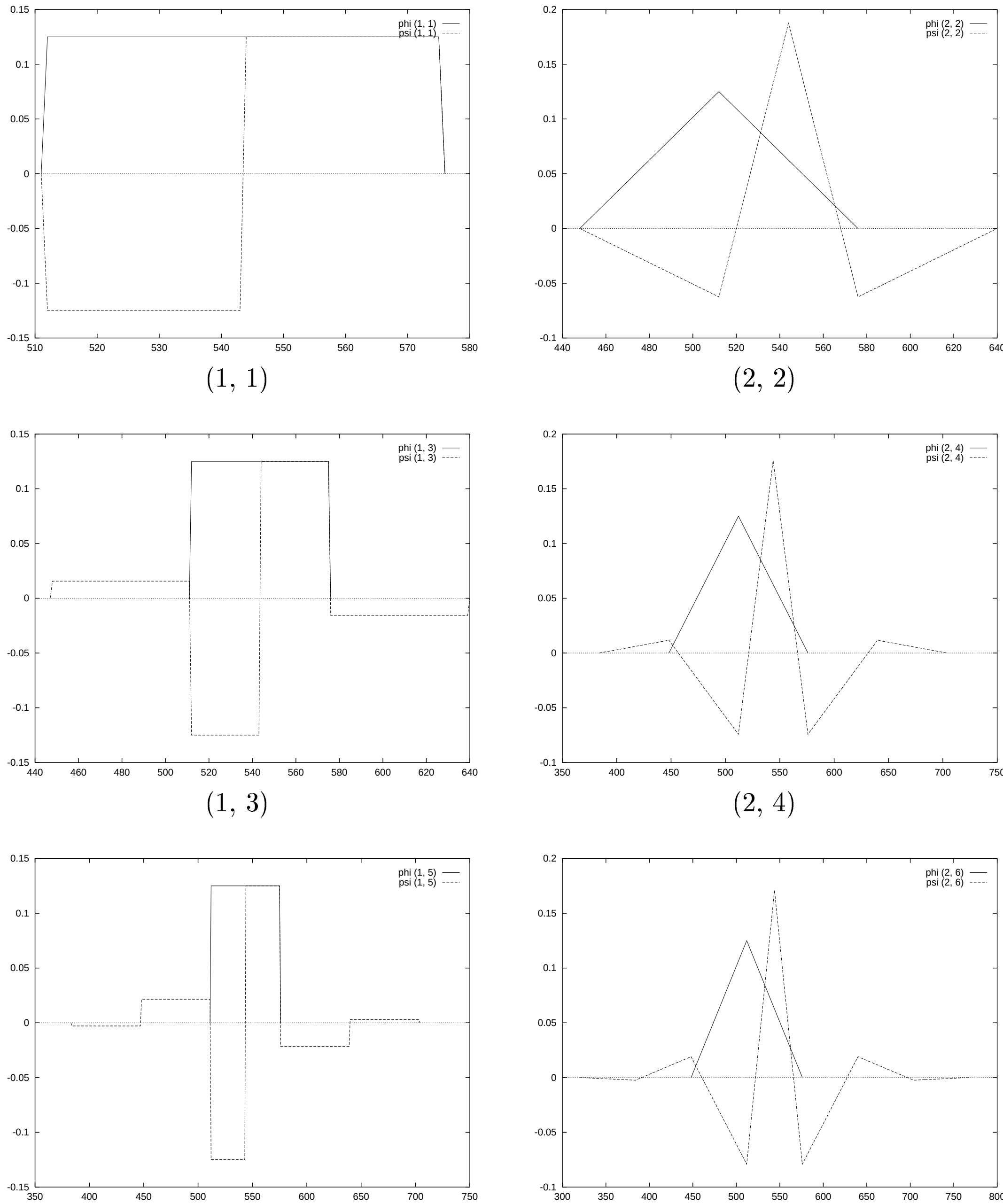
### 2.3 Example: Biorthogonal Cohen-Daubechies-Feauveau

This family of biorthogonal wavelets [2] has some interesting properties:

- The scaling function $\varphi(x)$ is always symmetric.
- The wavelet function $\psi(x)$ is always symmetric or antisymmetric.
- The wavelet filters are finite.
- The coefficients of the wavelet filters are of the form $\frac{z}{2^n}$, with $z$ integer and $n$ a natural number.

These wavelets are commonly classified by the number of vanishing moments they have: $(n, \tilde{n})$ means that the primal wavelet $\psi(x)$ — the synthesizing high-pass filter — has $n$ vanishing moments, while the dual wavelet $\widetilde{\psi}(x)$ — the analyzing high-pass filter — has $\tilde{n}$ vanishing moments.

The plots of some of the scaling functions $\varphi(x)$ and wavelet functions $\psi(x)$ of this family are shown in the figures 2, 3 and 4.

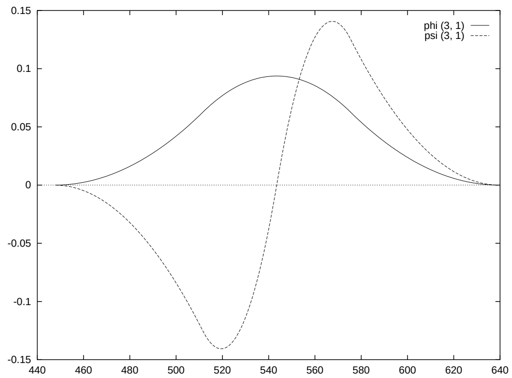



(1, 1) (2, 2)

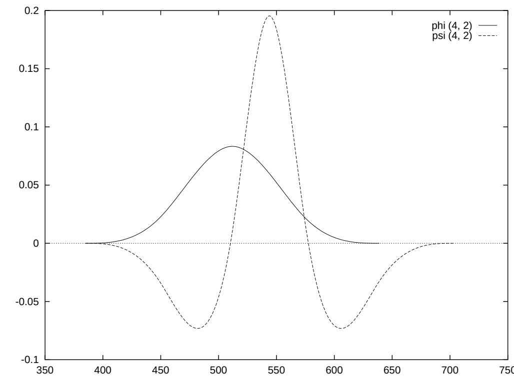
(1, 3) (2, 4)

(1, 5) (2, 6)

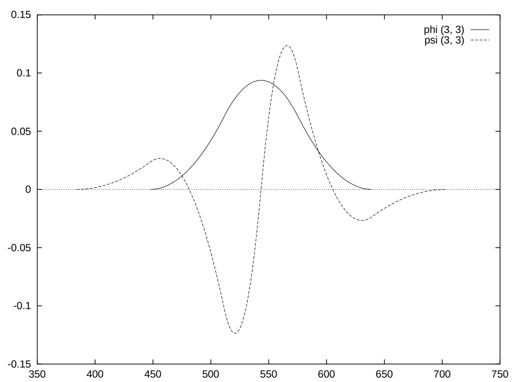
Figure 2: Cohen-Daubechies-Feauveau biorthogonal wavelets.



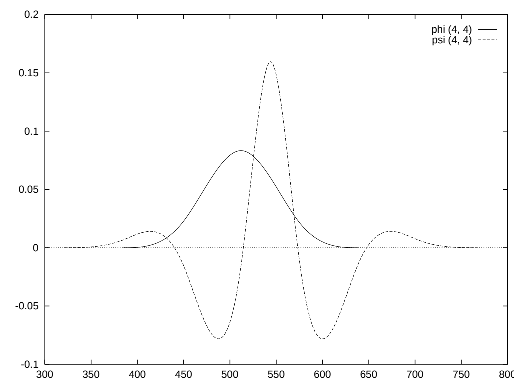

(3, 1)



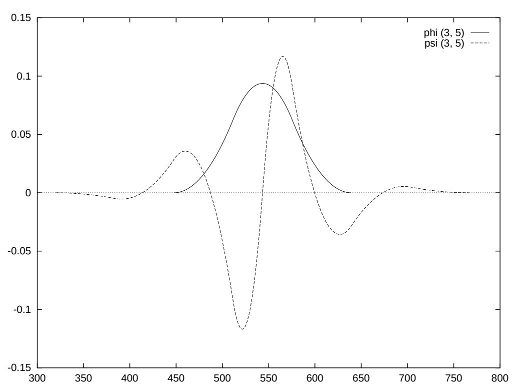

(4, 2)



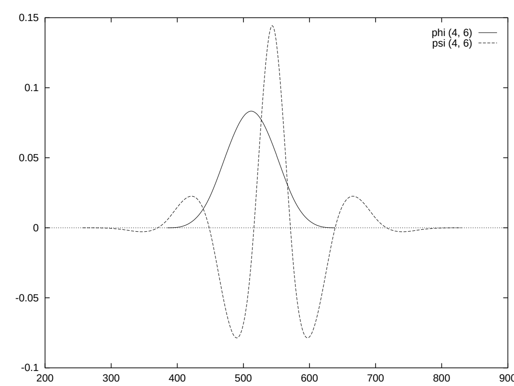

(3, 3)




(4, 4)




(3, 5)




(4, 6)

Figure 3: Cohen-Daubechies-Feauveau biorthogonal wavelets (cont'd).

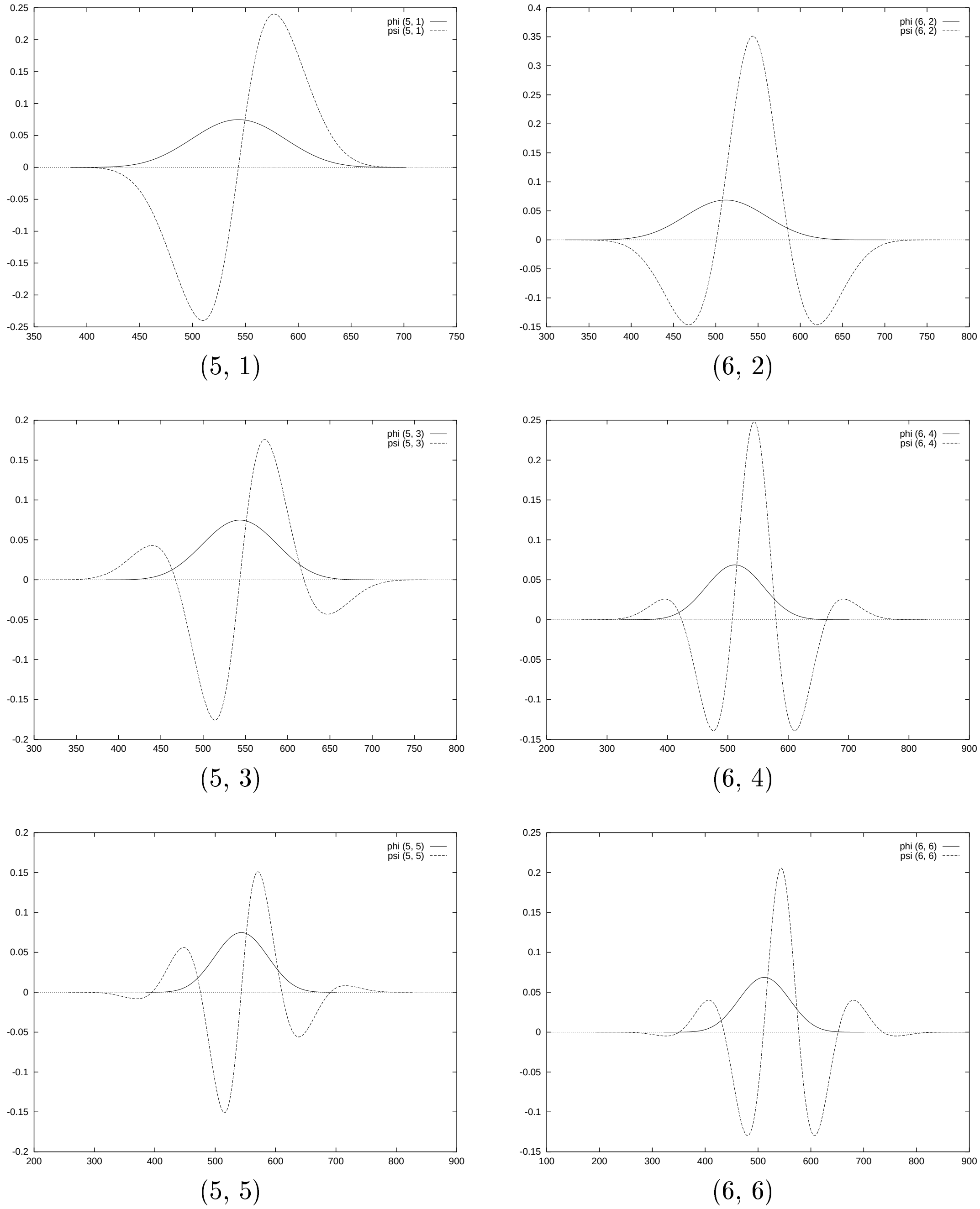



Figure 4: Cohen-Daubechies-Feauveau biorthogonal wavelets (cont'd).

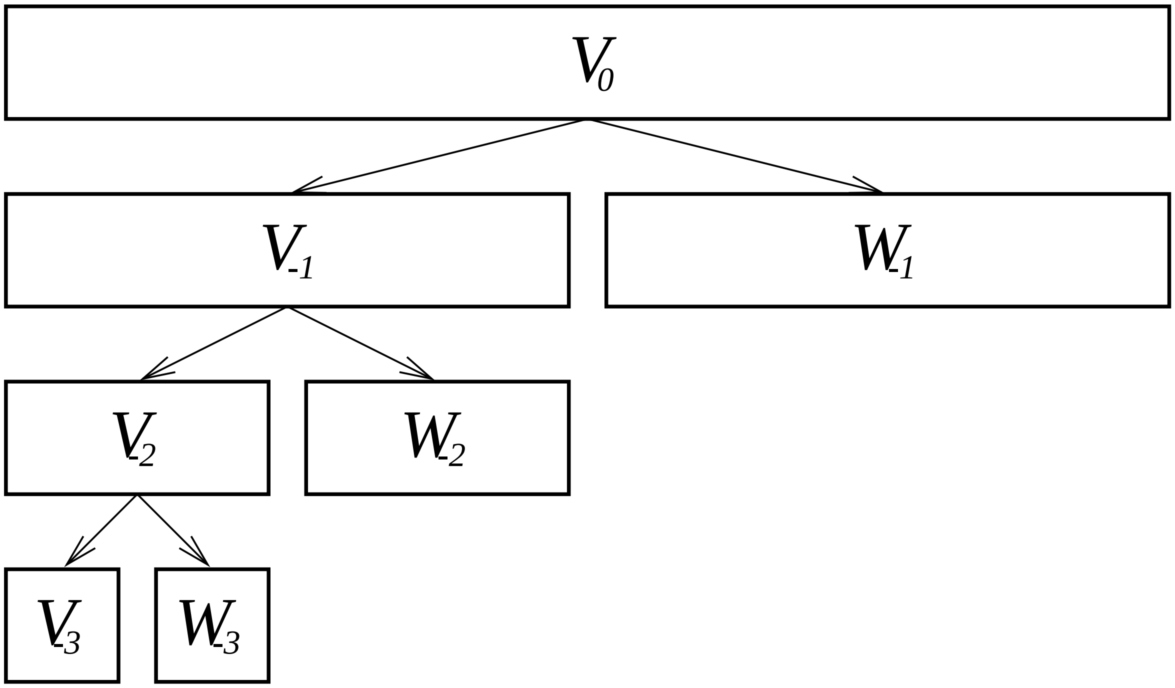



Figure 5: The decomposition tree for a wavelet transform.

### 2.4 Multi-Resolution

The signal $LP$ obtained by filtering with filter $h$ and subsampling resembles the original signal $S$ very much: it's a representation of $S$ at a lower resolution level. On the contrary, $HP$ contains the detail information that's lost by making the transition to that lower resolution level [5].

If we treat the low frequency information as a new signal and pass it through the same filter bank, and repeat these steps several times, we call this a wavelet transform. After some iterations we retain a very low frequency signal together with the detail information for the different resolution levels. This is shown by the decomposition tree in fig. 5. The blocks marked with $V_n$ represent the low frequency signal at resolution level $n$, while the $W_n$ represent the detail signal for the transition from resolution level $n$ to level $n+1$.

Because of the subsampling the length of the low pass signal is divided by two after every filter step. Thus the wavelet transform has a linear complexity ($\mathcal{O}(n)$) and can be executed fast.

### 2.5 The Mathematics behind the Wavelet Transform

The scaling function $\varphi(x)$ and the wavelet function $\psi(x)$ and their duals $\widetilde{\varphi}(x)$ and $\widetilde{\psi}(x)$ satisfy dilation equations which are:

$$\varphi(x) \quad = \quad \sqrt{2} \sum_k h_k \, \varphi(2x - k),$$

$$
\begin{aligned}
\psi(x) &= \sqrt{2}\sum_k g_k\,\varphi(2x-k),\\
\widetilde{\varphi}(x) &= \sqrt{2}\sum_k \widetilde{h}_k\,\widetilde{\varphi}(2x-k),\\
\widetilde{\psi}(x) &= \sqrt{2}\sum_k \widetilde{g}_k\,\widetilde{\varphi}(2x-k).
\end{aligned}
$$

The coefficients $h_k$, $g_k$, $\widetilde{h}_k$ and $\widetilde{g}_k$ represent the coefficients of the filters $h$, $g$, $\widetilde{h}$ and $\widetilde{g}$ respectively.

We define the basis functions

$$
\begin{aligned}
\varphi_{j,k}(x) &\equiv 2^{2/j}\varphi(2^j x-k),\\
\psi_{j,k}(x) &\equiv 2^{2/j}\psi(2^j x-k),
\end{aligned}
$$

with $j,k\in\mathbf{Z}$, i.e. $\varphi_{j,k}(x)$ is a translated and dilated version of $\varphi(x)$.

If we have a representation of a function at resolution level $j+1$:

$$
f(x)=\sum_k \lambda_{j+1,k}\,\varphi_{j+1,k}(x),
$$

then we can also decompose it into its low pass and high pass components:

$$
f(x)=\sum_l \lambda_{j,l}\,\varphi_{j,l}(x)+\sum_l \gamma_{j,l}\,\psi_{j,l}(x).
$$

The wanted coefficients $\lambda_{j,l}$ and $\gamma_{j,l}$ can be found by calculating the inner products with the dual basis functions:

$$
\begin{aligned}
\lambda_{j,l} &= \langle f,\widetilde{\varphi}_{j,l}\rangle,\\
\gamma_{j,l} &= \langle f,\widetilde{\psi}_{j,l}\rangle,
\end{aligned}
$$

with

$$
\langle f,g\rangle \equiv \int_{-\infty}^{\infty} f(x)\,g(x)\,dx.
$$

By using these equations we obtain the formulae for the Fast Wavelet Transform:

$$
\begin{aligned}
\lambda_{j,l} &= \sqrt{2}\sum_k \widetilde{h}_{k-2l}\,\lambda_{j+1,k},\\
\gamma_{j,l} &= \sqrt{2}\sum_k \widetilde{g}_{k-2l}\,\lambda_{j+1,k}.
\end{aligned}
$$

## 2.6 Properties of Wavelets

- Multi-resolution
- Locality in both the time/space domain and the frequency domain

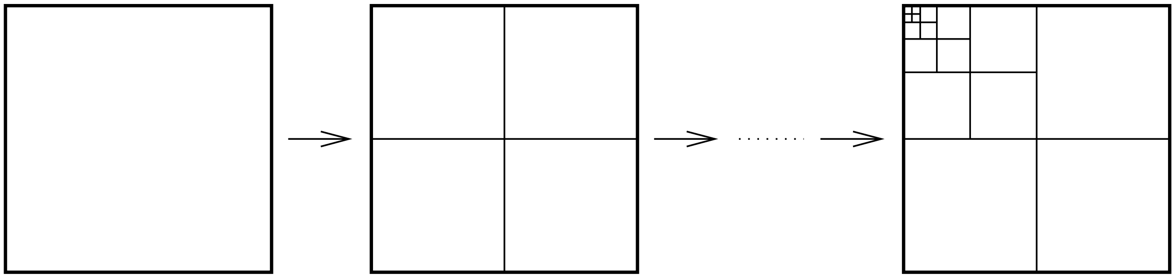


Figure 6: The two-dimensional wavelet transform.

- (bi)Orthogonality
- Smooth basis functions (controlled by vanishing moments)
- Good approximations (in $\mathbf{L}^2$)
- Fast transform algorithms ($\mathcal{O}n$)
- Stable decompositions

### 2.7 Higher Dimensions

One step of the wavelet transform of a signal with a dimension $n$ higher than 1 is performed by transforming each dimension of the signal independently. Afterwards the $n$-dimensional subband that contains the low pass part in all dimensions is transformed further. The two-dimensional case is shown in fig. 6.

### 2.8 A simple compression example

If we transform an image containing $N$ pixels using a wavelet transform, we have $N$ wavelet coefficients. Now we can consider a very simple image compression algorithm by replacing the $M$ smallest (in absolute value) wavelet coefficients by zero and comparing the result after reconstruction with the original image. A graph showing the Peak Signal to Noise Ratio versus the compression rate ($\frac{N}{N-M}$) for the well-known "Lena" image and some of the Cohen-Daubechies-Feauveau wavelet transforms is shown in fig. 7. Different test images yield different graphs, but the qualitative behavior of the graphs stay the same.

## 3 The Lifting Scheme

### 3.1 Introduction

The Lifting Scheme found its roots in a method to improve a given wavelet transform to obtain some specific properties. Later it was extended to a generic method to create so-called 'Second Generation' wavelets [9, 7, 6, 8].

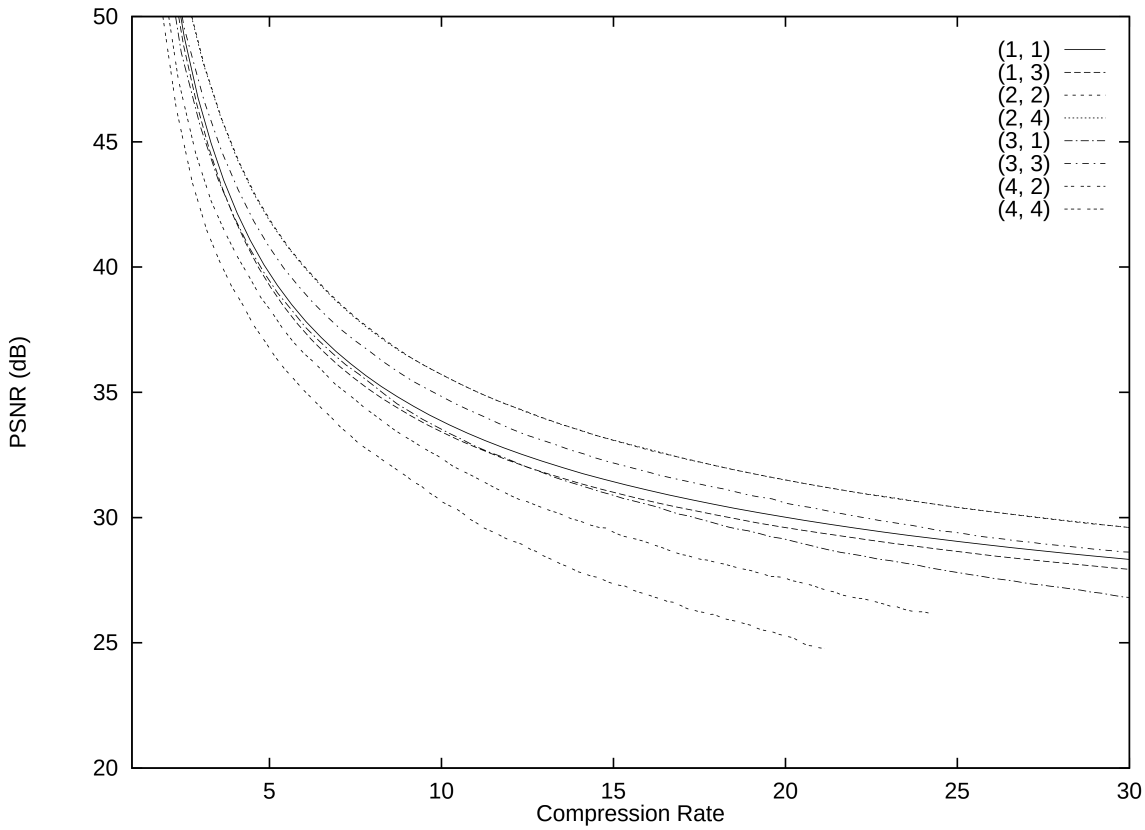



Figure 7: Peak Signal to Noise Ratio (PSNR) versus compression rate for some of the Cohen-Daubechies-Feauveau wavelet transforms.

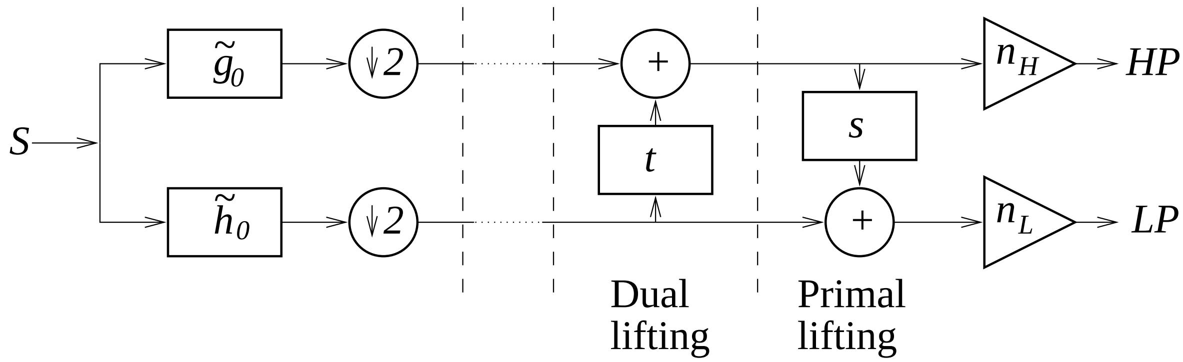



Figure 8: The Lifting Scheme.

The generic scheme is represented in fig. 8. Here the filters $\tilde{g}_0$, $\tilde{h}_0$ and the successive subsampling are the 'Lazy' wavelet transform. This transform doesn't do anything but split the original signal $S$ into two sequences containing the odd and the even samples. Afterwards the transform is 'lifted' to a transform with the wanted properties by using one or more of the filter operations $s$ and/or $t$. Finally the results are normalized by multiplying with the factors $n_H$ respectively $n_L$.

The inverse transform is immediately clear from looking at the figure: all we have to do is invert all steps. This is trivial here, unlike with the classical wavelet transform scheme.

Thus the goal is to find one or more lifting filters $s$ and/or $t$, and normalization factors $n_H$ and $n_L$, such that this scheme is equivalent to the one in fig. 1. One has proved that all classical wavelet transforms can be implemented using the Lifting Scheme [3].

Advantages of the Lifting Scheme over the classical wavelet transform are:

- Generic method
- Easier to understand (not introduced using the Fourier transform)
- Easier to implement
- Faster ($\times 2$, but still $\mathcal{O}(n)$)
- The inverse transform is easier to find
- The inverse transform has exactly the same complexity as the forward transform.
- Transforms signals with an arbitrary length (need not be $2^n$)
- In-place: requires less memory
- Transforms signals with a finite length (without extension of the signal)
- Can be used on arbitrary geometries

- Can be used on irregular samplings
- Can be extended for weighting functions
- All wavelet filters can be implemented using the Lifting Scheme
- Simple extension to an integer transform possible

The result of a wavelet transform using the Lifting Scheme contains all subband data in an interleaved form ('in-place' transform). However, for some applications the classical order ('Mallat' order [5], cfr. fig. 6) with separated subbands is better suited.

## 3.2 Algorithm

The flow of a wavelet transform algorithm using the Lifting Scheme looks like:

**1. Split:**

$$\begin{aligned} s_i &\leftarrow x_{2i} \\ d_i &\leftarrow x_{2i+1} \end{aligned}$$

**2. Lifting steps:** One or more steps $k$ of the form

**a. Primal lifting:**

$$s_i \leftarrow s_i - \sum_j l_j^k d_j$$

or

**b. Dual lifting:**

$$d_i \leftarrow d_i - \sum_j l_j^k s_j$$

**3. Normalization:**

$$\begin{aligned} s_i &\leftarrow n_L \cdot s_i \\ d_i &\leftarrow n_H \cdot d_i \end{aligned}$$

If the transform is normalized, we have $n_L \cdot n_H = 1$.

As you can see, the inverse transform easily follows from this algorithm: just reverse all steps.

### 3.3 Example: Biorthogonal Cohen-Daubechies-Feauveau

The analyzing filters for some of the popular biorthogonal Cohen-Daubechies-Feauveau wavelets are shown below, together with the lifting steps we obtained by using the technique described in [3]. Note that the coefficients of the filters are always of the form $\frac{z}{2^n}$, with $z \in \mathbf{Z}$ and $n \in \mathbf{N}$. This means that all divisions can be implemented using binary shifts. Unfortunately the coefficients of the lifting steps aren't always of this form.

#### 3.3.1 CDF $(1, x)$

$$
\begin{aligned}
\widetilde{g}_{(1,x)} &: \frac{\sqrt{2}}{2} \cdot \left( \; 1, -1 \; \right) \\
\widetilde{h}_{(1,1)} &: \frac{\sqrt{2}}{2} \cdot \left( \; 1, 1 \; \right) \\
\widetilde{h}_{(1,3)} &: \frac{\sqrt{2}}{16} \cdot \left( \; -1, 1, 8, 8, 1, -1 \; \right) \\
\widetilde{h}_{(1,5)} &: \frac{\sqrt{2}}{256} \cdot \left( \; 3, -3, -22, 22, 128, 128, 22, -22, -3, 3 \; \right)
\end{aligned}
$$

The lifting steps are:

$$
\begin{aligned}
d_i &\leftarrow d_i - s_i \\
s_i &\overset{(1,1)}{\leftarrow} s_i + \frac{1}{2} d_i \\
s_i &\overset{(1,3)}{\leftarrow} s_i - \frac{1}{16} \left( -d_{i-1} - 8d_i + d_{i+1} \right) \\
s_i &\overset{(1,5)}{\leftarrow} s_i - \frac{1}{256} \left( 3d_{i-2} - 22d_{i-1} - 128d_i + 22d_{i+1} - 3d_{i+2} \right)
\end{aligned}
$$

The normalization factors are:

$$
\begin{aligned}
n_L &= \sqrt{2}, \\
n_H &= \frac{\sqrt{2}}{2}.
\end{aligned}
$$

#### 3.3.2 CDF $(2, x)$

$$
\begin{aligned}
\widetilde{g}_{(2,x)} &: \frac{\sqrt{2}}{4} \cdot \left( \; 1, -2, 1 \; \right) \\
\widetilde{h}_{(2,2)} &: \frac{\sqrt{2}}{8} \cdot \left( \; -1, 2, 6, 2, -1 \; \right)
\end{aligned}
$$

$$
\begin{aligned}
\widetilde{h}_{(2,4)} \quad &: \quad \frac{\sqrt{2}}{128} \cdot \Big( \; 3, -6, -16, 38, 90, 38, -16, -6, 3 \; \Big) \\
\widetilde{h}_{(2,6)} \quad &: \quad \frac{\sqrt{2}}{1024} \cdot \Big( \; -5, 10, 34, -78, -123, 324, 700, 324, -123, -78, 34, 10, -5 \; \Big)
\end{aligned}
$$

The lifting steps are:

$$
\begin{aligned}
d_i \quad &\leftarrow \quad d_i - \frac{1}{2}\left(s_i + s_{i+1}\right) \\
s_i \quad &\overset{(2,2)}{\leftarrow} \quad s_i - \frac{1}{4}\left(-d_{i-1} - d_i\right) \\
s_i \quad &\overset{(2,4)}{\leftarrow} \quad s_i - \frac{1}{64}\left(3d_{i-2} - 19d_{i-1} - 19d_i + 3d_{i+1}\right) \\
s_i \quad &\overset{(2,6)}{\leftarrow} \quad s_i - \frac{1}{512}\left(-5d_{i-3} + 39d_{i-2} - 162d_{i-1} - 162d_i + 39d_{i+1} - 5d_{i+2}\right)
\end{aligned}
$$

The normalization factors are:

$$
\begin{aligned}
n_L \quad &= \quad \sqrt{2}, \\
n_H \quad &= \quad \frac{\sqrt{2}}{2}.
\end{aligned}
$$

### 3.3.3 CDF $(3, x)$

$$
\begin{aligned}
\widetilde{g}_{(3,x)} \quad &: \quad \frac{\sqrt{2}}{8} \cdot \Big( \; -1, 3, -3, 1 \; \Big) \\
\widetilde{h}_{(3,1)} \quad &: \quad \frac{\sqrt{2}}{4} \cdot \Big( \; -1, 3, 3, -1 \; \Big) \\
\widetilde{h}_{(3,3)} \quad &: \quad \frac{\sqrt{2}}{64} \cdot \Big( \; 3, -9, -7, 45, 45, -7, -9, 3 \; \Big) \\
\widetilde{h}_{(3,5)} \quad &: \quad \frac{\sqrt{2}}{512} \cdot \Big( \; -5, 15, 19, -97, -26, 350, 350, -26, -97, 19, 15, -5 \; \Big)
\end{aligned}
$$

The lifting steps are:

$$
\begin{aligned}
s_i \quad &\leftarrow \quad s_i - \frac{1}{3}d_{i-1} \\
d_i \quad &\leftarrow \quad d_i - \frac{1}{8}\left(9s_i + 3s_{i+1}\right) \\
s_i \quad &\overset{(3,1)}{\leftarrow} \quad s_i + \frac{4}{9}d_i \\
s_i \quad &\overset{(3,3)}{\leftarrow} \quad s_i - \frac{1}{36}\left(-3d_{i-1} - 16d_i + 3d_{i+1}\right)
\end{aligned}
$$

$$s_i \overset{(3,5)}{\leftarrow} s_i - \frac{1}{288}\left(5d_{i-2} - 34d_{i-1} - 128d_i + 34d_{i+1} - 5d_{i+2}\right)$$

The normalization factors are:

$$\begin{aligned} n_L &= \frac{3\sqrt{2}}{2}, \\ n_H &= \frac{\sqrt{2}}{3}. \end{aligned}$$

### 3.3.4 CDF $(4, x)$

$$\begin{aligned}
\widetilde{g}_{(4,x)} &: \frac{\sqrt{2}}{16} \cdot \left( -1, 4, -6, 4, -1 \right) \\
\widetilde{h}_{(4,2)} &: \frac{\sqrt{2}}{32} \cdot \left( 3, -12, 5, 40, 5, -12, 3 \right) \\
\widetilde{h}_{(4,4)} &: \frac{\sqrt{2}}{512} \cdot \left( -10, 40, -2, -192, 140, 560, 140, -192, -2, 40, -10 \right) \\
\widetilde{h}_{(4,6)} &: \frac{\sqrt{2}}{8192} \cdot \left( \begin{array}{l} 35, -140, -55, 920, -557, -2932, 2625, 8400, \\ 2625, -2932, -557, 920, -55, -140, 35 \end{array} \right)
\end{aligned}$$

The lifting steps are:

$$\begin{aligned}
s_i &\leftarrow s_i - \frac{1}{4}\left(d_{i-1} + d_i\right) \\
d_i &\leftarrow d_i - \left(s_i + s_{i+1}\right) \\
s_i &\overset{(4,2)}{\leftarrow} s_i - \frac{1}{16}\left(-3d_{i-1} - 3d_i\right) \\
s_i &\overset{(4,4)}{\leftarrow} s_i - \frac{1}{128}\left(5d_{i-2} - 29d_{i-1} - 29d_i + 5d_{i+1}\right) \\
s_i &\overset{(4,6)}{\leftarrow} s_i - \frac{1}{4096}\left(-35d_{i-3} + 265d_{i-2} - 998d_{i-1} - 998d_i + 265d_{i+1} - 35d_{i+2}\right)
\end{aligned}$$

The normalization factors are:

$$\begin{aligned} n_L &= 2\sqrt{2}, \\ n_H &= \frac{\sqrt{2}}{4}. \end{aligned}$$

### 3.3.5 CDF $(5, x)$

$$
\begin{array}{lcl}
\widetilde{g}_{(5,x)} & : & \dfrac{\sqrt{2}}{32} \cdot \left( \; 1, -5, 10, -10, 5, -1 \; \right) \\
\widetilde{h}_{(5,1)} & : & \dfrac{\sqrt{2}}{16} \cdot \left( \; 3, -15, 20, 20, -15, 3 \; \right) \\
\widetilde{h}_{(5,3)} & : & \dfrac{\sqrt{2}}{128} \cdot \left( \; -5, 25, -26, -70, 140, 140, -70, -26, 25, -5 \; \right) \\
\widetilde{h}_{(5,5)} & : & \dfrac{\sqrt{2}}{4096} \cdot \left( \begin{array}{l} 35, -175, 120, 800, -1357, -1575, 4200, \\ 4200, -1575, -1357, 800, 120, -175, 35 \end{array} \right)
\end{array}
$$

The lifting steps are:

$$
\begin{array}{lcl}
d_i & \leftarrow & d_i - \dfrac{1}{5} s_i \\
s_i & \leftarrow & s_i - \dfrac{1}{24} \left( 15 d_{i-1} + 5 d_i \right) \\
d_i & \leftarrow & d_i - \dfrac{1}{10} \left( 15 s_i + 9 s_{i+1} \right) \\
s_i & \overset{(5,1)}{\leftarrow} & s_i + \dfrac{1}{3} d_i \\
s_i & \overset{(5,3)}{\leftarrow} & s_i - \dfrac{1}{72} \left( -5 d_{i-1} - 24 d_i + 5 d_{i+1} \right) \\
s_i & \overset{(5,5)}{\leftarrow} & s_i - \dfrac{1}{2304} \left( 35 d_{i-2} - 230 d_{i-1} - 768 d_i + 230 d_{i+1} - 35 d_{i+2} \right)
\end{array}
$$

The normalization factors are:

$$
\begin{array}{lcl}
n_L & = & 3\sqrt{2}, \\
n_H & = & \dfrac{\sqrt{2}}{6}.
\end{array}
$$

### 3.3.6 CDF $(6, x)$

$$
\begin{array}{lcl}
\widetilde{g}_{(6,x)} & : & \dfrac{\sqrt{2}}{64} \cdot \left( \; 1, -6, 15, -20, 15, -6, 1 \; \right) \\
\widetilde{h}_{(6,2)} & : & \dfrac{\sqrt{2}}{64} \cdot \left( \; -5, 30, -56, -14, 154, -14, -56, 30, -5 \; \right) \\
\widetilde{h}_{(6,4)} & : & \dfrac{\sqrt{2}}{2048} \cdot \left( \; 35, -210, 330, 470, -1827, 252, 3948, 252, -1827, 470, 330, -210, 35 \; \right) \\
\widetilde{h}_{(6,6)} & : & \dfrac{\sqrt{2}}{16384} \cdot \left( \begin{array}{l} -63, 378, -476, -1554, 4404, 1114, -13860, 4158, 28182, \\ 4158, -13860, 1114, 4404, -1554, -476, 378, -63 \end{array} \right)
\end{array}
$$

The lifting steps are:

$$\begin{aligned}
d_i &\leftarrow d_i - \frac{1}{6}\left(s_i + s_{i+1}\right) \\
s_i &\leftarrow s_i - \frac{1}{16}\left(9d_{i-1} + 9d_i\right) \\
d_i &\leftarrow d_i - \frac{1}{3}\left(4s_i + 4s_{i+1}\right) \\
s_i &\overset{(6,2)}{\leftarrow} s_i - \frac{1}{32}\left(-5d_{i-1} - 5d_i\right) \\
s_i &\overset{(6,4)}{\leftarrow} s_i - \frac{1}{1024}\left(35d_{i-2} - 195d_{i-1} - 195d_i + 35d_{i+1}\right) \\
s_i &\overset{(6,6)}{\leftarrow} s_i - \frac{1}{8192}\left(-63d_{i-3} + 469d_{i-2} - 1686d_{i-1} - 1686d_i - +469d_{i+1} - 63d_{i+2}\right)
\end{aligned}$$

The normalization factors are:

$$\begin{aligned}
n_L &= 4\sqrt{2}, \\
n_H &= \frac{\sqrt{2}}{8}.
\end{aligned}$$

### 3.4 The integer wavelet transform

In many applications (e.g. image compression and processing) the input data consists of integer samples. Unfortunately all of the above transforms assume the input samples are floating point values. They return floating point values as wavelet coefficients, even if the input values actually were integer. Rounding the floating point values to integer values doesn't help because then we will loose the perfect reconstruction feature.

Fortunately the lifting scheme can be easily modified to a transform that maps integers to integers and that is reversible, and thus allows a perfect reconstruction [1]. This will be done by adding some rounding operations, at the expense of introducing a non-linearity in the transform.

A lifting step basically looks like

$$x_i \leftarrow x_i - \frac{1}{a}\sum_j b_j y_j.$$

Here we assume that $a, b_j \in \mathbf{Z}$, i.e. they are integers. This is true for various wavelet transforms, among which the Cohen-Daubechies-Feauveau biorthogonal wavelets.

This lifting step can be modified in one of the following ways. A rounding of $x$ to the nearest integer value will be indicated by $\{x\}$.

**Full rounding** The result of the division by $a$ is rounded:

$$\tilde{x}_i \leftarrow x_i - \left\{\frac{\sum_j b_j y_j}{a}\right\}.$$

Here $\widetilde{x}_i$ is an integer approximation to $x_i$.

**Without rounding** We avoid the division by $a$ by multiplying the other terms with $a$:

$$\widetilde{x}_i \leftarrow a x_i - \sum_j b_j y_j.$$

Here $\widetilde{x}_i = ax$, and thus the dynamic range of the wavelet coefficients will increase. This has to be taken into account in later steps. Note that in this case no real rounding is performed, and thus this can be considered to be an 'exact' implementation of the floating point version, yielding integers.

**Mixed form** We combine the rounding and multiplication steps of both methods:

$$\widetilde{x}_i \leftarrow a_1 x_i - \left\{ \frac{\sum_j b_j y_j}{a_2} \right\},$$

with

$$\begin{aligned} a_1, a_2 &\in \mathbf{Z}, \\ a_1 \cdot a_2 &= a. \end{aligned}$$

This variant can be used if we want to have more control over the dynamic range of the resulting $\widetilde{x}_i$.

Note that in all of the three cases above the modified lifting step is still reversible, and thus the perfect reconstruction feature is still present.

If $a, b_j \notin \mathbf{Z}$, we can still use the 'full rounding' method to obtain a transform that maps integers to integers.

### Examples

**CDF** $(1, x)$ This is the well-known Haar transform. The lifting steps

$$\begin{aligned} d_i &\leftarrow d_i - s_i, \\ s_i &\leftarrow s_i + \frac{1}{2} d_i, \end{aligned}$$

can be changed into

$$\begin{aligned} d_i &\leftarrow d_i - s_i, \\ s_i &\leftarrow s_i + \left\{ \frac{1}{2} d_i \right\}, \end{aligned}$$

yielding a transform that maps integers to integers, if we forget about the normalization factors.

**CDF** $(2, x)$ The lifting steps

$$\begin{aligned} d_i &\leftarrow d_i - \frac{1}{2}(s_i + s_{i+1}), \\ s_i &\leftarrow s_i - \frac{1}{4}(-d_{i-1} - d_i), \end{aligned}$$

can be changed into

$$\begin{aligned} d_i &\leftarrow 2d_i - s_i - s_{i+1}, \\ s_i &\leftarrow s_i + \left\{ \frac{d_{i-1} + d_i}{8} \right\}. \end{aligned}$$

Note that the divider in the second lifting step is changed from 4 to 8 to compensate for the multiplication with 2 in the first step. An alternative, with rounding in both steps, would be:

$$\begin{aligned} d_i &\leftarrow d_i - \left\{ \frac{s_i + s_{i+1}}{2} \right\}, \\ s_i &\leftarrow s_i + \left\{ \frac{d_{i-1} + d_i}{4} \right\}. \end{aligned}$$

## 3.5 Bounds of wavelet coefficients and bit growth

Ideally we would like the normalization factor for the low-pass data to be $\sqrt{2}$ in all cases. Indeed, then the dynamic range of the low-pass data stays the same if we forget about the normalization factor. If the input samples are 8-bit values, the low-pass data after applying a wavelet transform step (the 'mean' values) will be 8-bit values too, i.e. there is no bit growth in the low-pass band.

Of course we will have to 'remember' the accumulated normalization factors for every subband if we want to process the wavelet transformed data later. One way to do this is to consider each wavelet coefficient (i.e. the value after filtering) to be the mantissa $m$ of a floating point number in base $\sqrt{2}$. The exponent $e$ of this floating point number is $\log_{\sqrt{2}}$ of the accumulated normalization factor, which depends on the subband. Thus the real normalized value of each coefficient becomes:

$$x = m \cdot \sqrt{2}^e.$$

A special and simpler case of this is the two-dimensional wavelet transform where we always apply the transform to both the rows and the columns of a matrix. Since $\sqrt{2} \times \sqrt{2} = 2$, the normalization factor becomes 2 and the base of our floating point numbers can be 2 also.

**Example**

For CDF $(2, 2)$, the lifting steps we use are

$$d_i \leftarrow 2d_i - s_i - s_{i+1},$$

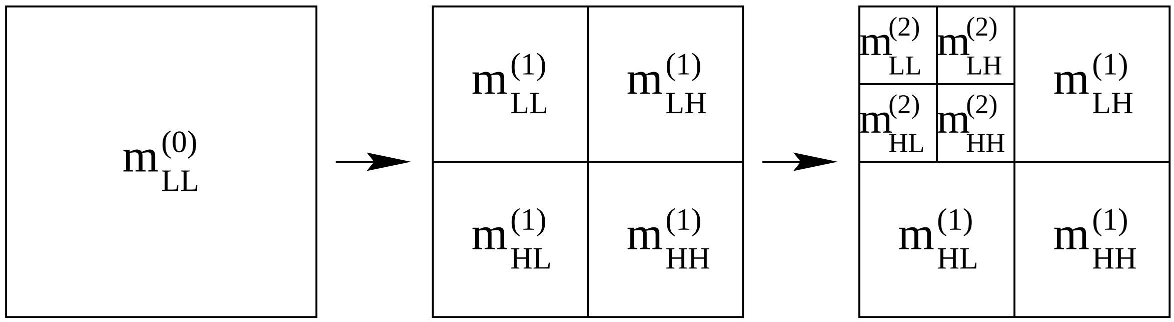



Figure 9: Two steps of the two-dimensional wavelet transform.

$$s_i \quad \leftarrow \quad s_i + \left\{ \frac{d_{i-1} + d_i}{8} \right\}.$$

Because of the multiplication by 2 in the first lifting step 1, the $d_i$ will be twice as large as normal and the normalization factors become

$$\begin{aligned} n_L &= \sqrt{2}, \\ n_H &= \frac{\sqrt{2}}{4}. \end{aligned}$$

In two dimensions, we will have 4 subbands $(LL, HL, LH, HH)$ after one transform step, with normalizations factors:

$$\begin{aligned} n_{LL} &= n_L \cdot n_L = 2 = \left(\sqrt{2}\right)^2 \\ n_{LH} &= n_L \cdot n_H = \frac{1}{2} = \left(\sqrt{2}\right)^{-2} \\ n_{HL} &= n_H \cdot n_L = \frac{1}{2} = \left(\sqrt{2}\right)^{-2} \\ n_{HH} &= n_H \cdot n_H = \frac{1}{8} = \left(\sqrt{2}\right)^{-6} \end{aligned}$$

Applying two wavelet transform steps, as shown in fig. 9, gives the following results. $x_S^{(l)}$ and $m_S^{(l)}$ represent a coefficient value respectively a mantissa in subband $S$ at decomposition level $l$.

$$
\begin{array}{llclcll}
\textbf{Level 0} & LL & : & x_{LL}^{(0)} & = & m_{LL}^{(0)} & = & m_{LL} \cdot 2^0 \\[2ex]
\textbf{Level 1} & LL & : & x_{LL}^{(1)} & = & m_{LL}^{(1)} \cdot \left(\sqrt{2}\right)^2 & = & m_{LL}^{(1)} \cdot 2^1 \\
 & LH & : & x_{LH}^{(1)} & = & m_{LH}^{(1)} \cdot \left(\sqrt{2}\right)^{-2} & = & m_{LH}^{(1)} \cdot 2^{-1} \\
 & HL & : & x_{HL}^{(1)} & = & m_{HL}^{(1)} \cdot \left(\sqrt{2}\right)^{-2} & = & m_{HL}^{(1)} \cdot 2^{-1} \\
 & HH & : & x_{HH}^{(1)} & = & m_{HH}^{(1)} \cdot \left(\sqrt{2}\right)^{-6} & = & m_{HH}^{(1)} \cdot 2^{-3} \\[2ex]
\textbf{Level 2} & LL & : & x_{LL}^{(2)} & = & m_{LL}^{(2)} \cdot \left(\sqrt{2}\right)^2 \cdot \left(\sqrt{2}\right)^2 & = & m_{LL}^{(2)} \cdot 2^2 \\
 & LH & : & x_{LH}^{(2)} & = & m_{LH}^{(2)} \cdot \left(\sqrt{2}\right)^2 \cdot \left(\sqrt{2}\right)^{-2} & = & m_{LH}^{(2)} \cdot 2^0 \\
 & HL & : & x_{HL}^{(2)} & = & m_{HL}^{(2)} \cdot \left(\sqrt{2}\right)^2 \cdot \left(\sqrt{2}\right)^{-2} & = & m_{HL}^{(2)} \cdot 2^0 \\
 & HH & : & x_{HH}^{(2)} & = & m_{HH}^{(2)} \cdot \left(\sqrt{2}\right)^2 \cdot \left(\sqrt{2}\right)^{-6} & = & m_{HH}^{(2)} \cdot 2^{-2}
\end{array}
$$

## References


[1] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. Technical report, Department of Mathematics, Princeton University, 1996.

[2] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 45:485–560, 1992.

[3] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. Technical report, Bell Laboratories, Lucent Technologies, 1996.

[4] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36(3):377–412, 1994.

[5] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 11(7):674–693, 1989.

[6] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics, (SIGGRAPH '95 Proceedings)*, 1995.

[7] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, (to appear).

[8] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[9] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996.